Ronak Patel

GT ID: 902780243

GT Username: rpatel88

CS 7641: Machine Learning Assignment #1

Spam Messages and Letter Recognition Using Supervised Learning Techniques

Dataset Overview

Spam Dataset:

Classifying spam data has been becoming increasingly important as the use of the Internet has scaled. Spam messages for advertising-related, adult-related, and unwanted marketing messages have been rising annually. Consequently, countermeasures such as anti-spam and built-in spam detection in popular email hosts have been flagging and classifying spam appropriately. It is for this reason, I would like to look deeper into how spam messages can be classified.

The spam dataset analyzed in this paper was selected from the UCI data repository. The dataset uses 57 attributes over 4601 instances (also referred to as samples) to characterize spam data for a user named George Forman. Furthermore, 54 of the 57 attributes (also referred to as features) are a measure of the frequencies of a particular word or character in an email. To name a few, words such as "George, make, address, all, and money" and characters such as "$, #, !" are all counted and calculated as a percentage of the words and characters over the entire email. From this binary classification dataset, 39.4% of emails are truly spam, while 60.6% are not spam. Given the large number of attributes relative to the number of instances it will be interesting to see if there are any dimensionality, overfitting, or underfitting problems.

Letter Recognition Dataset:

Letter recognition proves to be an interesting topic for natural language processing and computer vision. Recognizing text based on pixel statistics can become a power solution when transcribing handwritten information or books into a digital format. The goal with this dataset is to analyze if learners can classify these alphabetic images correctly.

The letter recognition dataset was taken from the UCI dataset repository. It is comprised of images of 26 capital letters from the English alphabet, which serve as the classes in the dataset. The character images are based on 20 different fonts that are randomly distorted to produce 20,000 unique instances. There are 16 total attributes in this data set and each of them are associated either a measure of statistical moments (correlation, variance, mean, etc.) or edge counts of the pixel information. Given these attributes, it will be interesting to see if any overfitting occurs during analysis since the majority of the attributes are statistics derived from the same pixel information.

Methodology

Each dataset was processed using 5 different supervised machine learning leaners adapted from Python's Sciki-learn library. These learners include:

- Decision Tree (DT) with pruning using Gini and Entropy information gain indexes
- Boosted (using AdaBoost) DT with pruning using Gini and Entropy information gain indexes
- K-Nearest Neighbors (KNN) using Euclidean, Manhattan, and Chebyshev distances
- Artificial Neural Network (ANN) using an MLP classifier with a varying nodes and layer count
- Support Vector Machines (SVM) using linear and a radial basis function (RBF) kernels

Training and test data were split up 70% and 30% accordingly. 5-fold cross validation was used in the training and test data to reduce the possibility of overfitting and overfitting while collecting data built on averages. Hyperparameters were optimized using a grid search to test combinations of hyperparameters and ranked based on accuracy.

Results

Decision Tree Learner

The decision tree learner was trained using both, Entropy and a Gini Impurity Indexes as a splitting criterion. Additionally, the class weight hyperparameter was tested as either balanced and unbalanced When the balanced hyperparameter was used, it will give more importance to correcting error in the smaller class sizes. Tables 1.1 and 1.2 below compare the results of the best scores (indicated by Test Rank) respective to the Gini and Entropy impurity index hyperparameters.

Table 1.1. Spam Data - DT Hyperparameters and Scores

Test Rank	Split Criterion	Tree Max Depth	Class Weight	Mean Test score	Mean Training Score
1	Gini	26	Balanced	91.29%	93.82%
27	Entropy	14	None	91.14%	94.37%

Table 1.2. Letter Recognition Data - DT Hyperparameters and Scores

Test Rank	Split Criterion	Tree Max Depth	Class Weight	Mean Test score	Mean Training Score
1	Entropy	14	Balanced	82.74%	88.38%
75	Gini	26	None	82.35%	88.38%

The tables above show that accuracy scores between the Gini and Entropy indexes, as well as balanced and unbalanced class weights, are almost identical. In a case where one may be concerned with depth

size and the generalizability of the tree, the model complexity charts, shown in Figures 1.1 and 1.2, illustrate that the tradeoff between performance and tree max depth is negligible where the curves begin to flatten. The tree can be further pruned in each set of data with negligible compromise on accuracy if needed.

However, it should be noted that the accuracy of the decision tree learner for the letter recognition dataset was relatively poor with respect to the other learners (to be covered subsequently). It can be suspected that there is high variability in the feature values of the dataset, and as a result, low information gain when calculating split values. Ultimately, this comes down to the decision tree algorithm itself, it does not consider interdependency of over variables or the target variable when creating the model, resulting in decreased accuracy in this case.

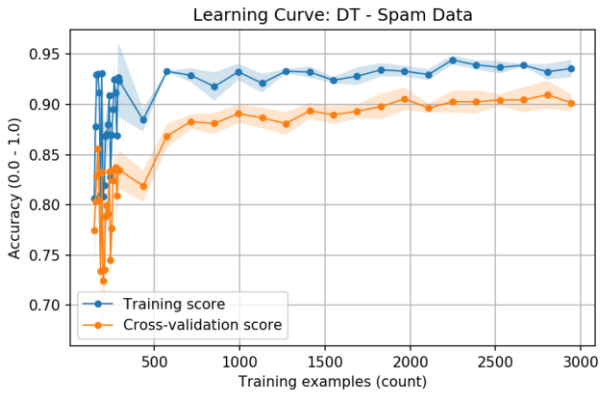

Figure 1.1 Spam Data DT Learning Curve

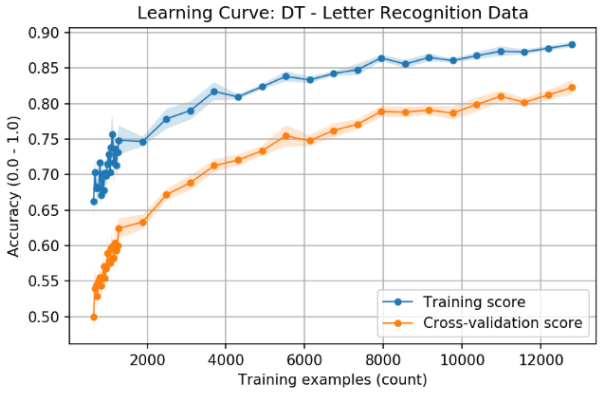

Figure 1.2 Letter Recognition Data DT Learning Curve

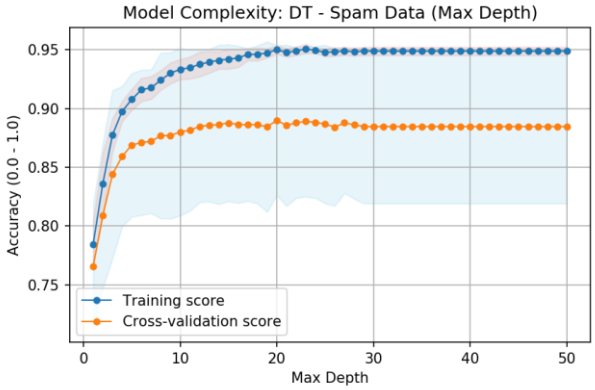

Figure 1.3 Spam Data DT Model Complexity Curve

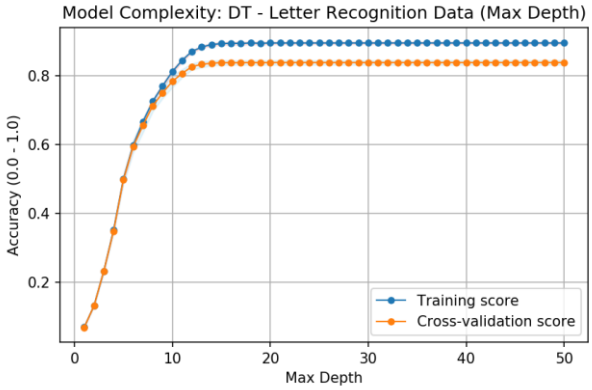

Figure 1.4 Letter Recognition Data DT Model Complexity Curve

Decision Tree with Adaboost Learner

The Adaboost algorithm uses an ensemble of weak decision tree learners to improve classification accuracy. Weak learners are characterized by having an accuracy metric above random chance (greater

than 50%), and these learners generally have high bias and/or high variance within the model. It is the weighted calculation of each of the weak learners' output that make this a strong classification algorithm. Hyperparameters in the ensemble, such as, the number of trees and the depth of each tree (also characterized by n_estimators) were tuned to achieve maximum accuracy. The outputs of these metrics can be seen in Tables 2.1 and 2.2 below.

Table 2.1. Spam Data – DT AdaBoost Hyperparameters and Scores

Rank	Max Tree Depth	Learning Rate	Number of Estimators	Mean Test Score	Mean Training Score
1	10	0.08	80	94.33%	96.70%
2	10	0.08	90	94.31%	96.64%
3	10	0.08	100	94.29%	96.77%

Table 2.2. Letter Recognition Data - DT AdaBoost Hyperparameters and Scores

Rank	Max Tree Depth	Learning Rate	Number of Estimators	Mean Test Score	Mean Training Score
1	10	0.16	80	93.68%	98.99%
1	10	0.16	90	93.68%	98.99%
1	10	0.16	100	93.68%	98.99%

Tables 2.1 and 2.2 above and the learning curves below in Figures 2.1 and 2.2 indicate that the application of boosting to the decision tree resulted in a reduction of bias and an increase in accuracy. Furthermore, the learning curves also show that the effect on variance was negligible since the accuracy of the scores have increased overall but the distance between the training set and the cross validated set accuracies have remained fairly constant. The model complexity curves in Figures 2.3 and 2.4 below show that increasing the number of weak learners only increases accuracy until a threshold is met and increasing learners yields no performance improvement. This is likely because adding more weak learners doesn't provide any additional useful information to change split criterion/values once the threshold is met. Finally, it can be seen from Figures 2.5 and 2.6 that there is a decrease in effectiveness when the learning rate increases beyond the 10^{-1} magnitude. In this case, the learner is trying to learn between interations too eagerly, resulting in underfitted data.

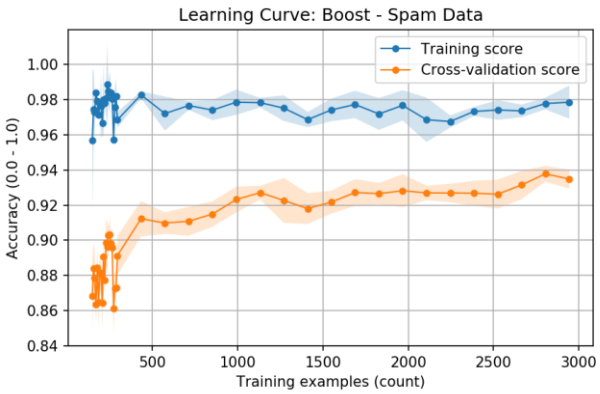

Figure 2.1 Spam Data Boosted DT Learning Curve

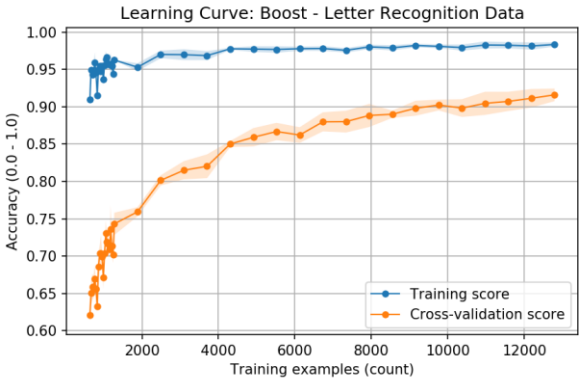

Figure 2.2 Letter Recognition Data Boosted DT Learning Curve

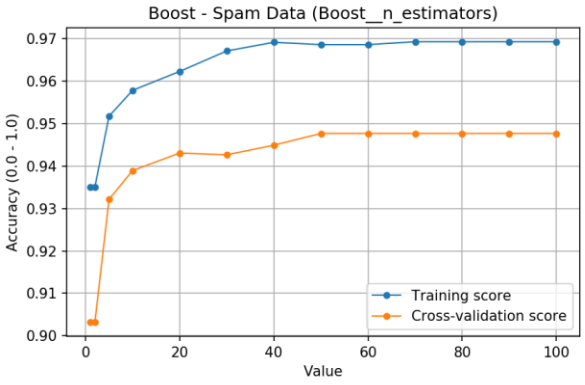

Figure 2.3 Spam Data N-Neighbor Model Complexity

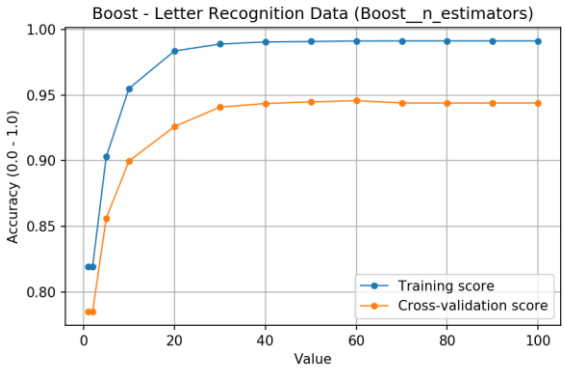

Figure 2.4 Letter Recognition N-Neighbor Model Complexity

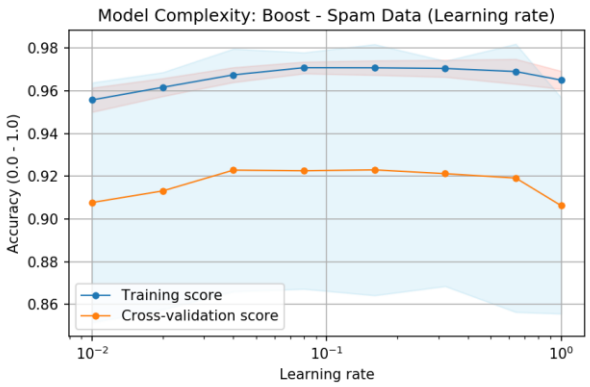

Figure 2.5 Spam Data Learning Rate Model Complexity

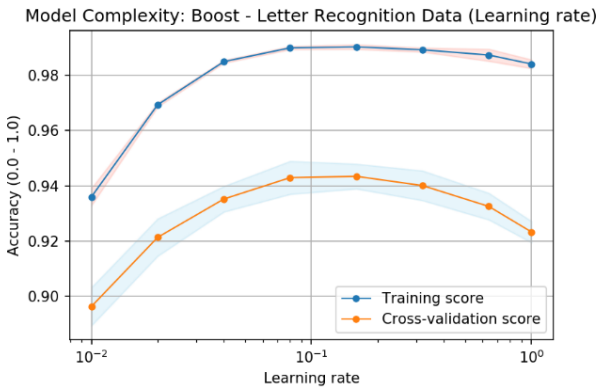

Figure 2.6 Letter Recognition Learning Rate Model Complexity

K-Nearest Neighbors (KNN) Learner

KNN classification algorithm was implemented using differing distance metrics within its hyperparameters. Namely, these are Euclidean, Manhattan, and Chebyshev distances. The best output of each were put in Tables 3.1 and 3.2 below along with additional data for the varying K hyperparameter.

Table 3.1. Spam Data – KNN Hyperparameters and Scores

Rank	K (nearest neighbors)	Distance parameter	Mean test score	Mean Training Score
1	1	Euclidean	89.88%	99.96%
2	1	Manhattan	89.61%	99.96%
3	7	Manhattan	89.23%	91.91%
4	7	Euclidean	89.20%	91.46%
5	1	Chebyshev	89.07%	99.96%

Table 3.2. Letter Recognition – KNN Hyperparameters and Scores

Rank	K (nearest neighbors)	Distance parameter	Mean test score	Mean Training Score
1	1	Euclidean	94.53%	100.00%
2	1	Manhattan	94.37%	100.00%
3	4	Manhattan	94.09%	97.11%
4	7	Manhattan	93.86%	96.33%
5	4	Euclidean	93.64%	96.86%
17	1	Chebyshev	91.01%	100.00%

The tables above indicate that K = 1 had the best performance in both datasets, however, values K = 4 and K = 7 also proved to be effective with negligible effect on accuracy. Both model complexity curves, shown in Figures 3.3 and 3.4, illustrate that generally as the K-value neighbor count increased, performance decreased. As the K-value increases, the scope of the classification parameter (K) increases, which will result in over-generalizing and underfitting of the training data. It can also be seen that the differing Euclidean, Manhattan, and Chebyshev distance measures had negligible effect on accuracy performance of the spam data, but generally performed worse on the letter recognition data in comparison to Euclidean and Manhattan distance parameters. The Chebyshev distance metric only uses the max distance across one dimension between data points, so distance may be getting undermined when there is higher variability across dimensions in the letter recognition data. Finally, the learning curves in Figures 3.1 and 3.2 do not show any odd behavior, indicating that the training model was an acceptable fit for the test data.

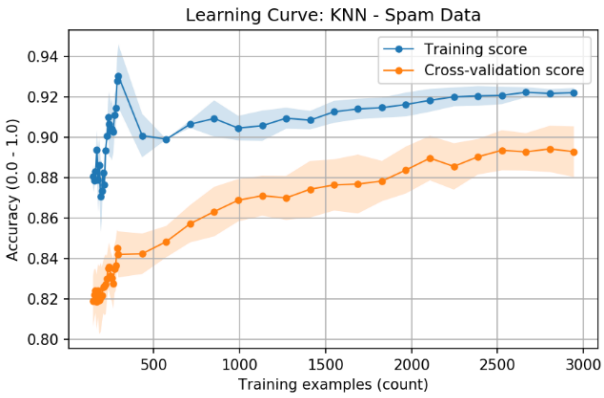

Figure 3.1 Spam Data KNN Learning Curve

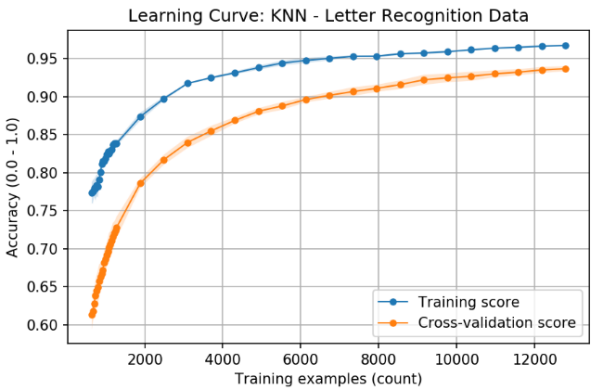

Figure 3.2 Letter Recognition Data KNN Learning Curve

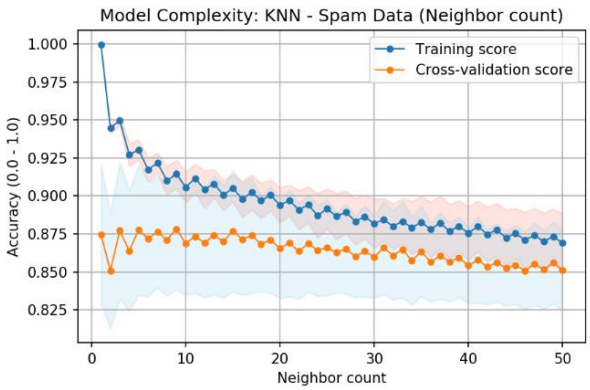

Figure 3.3 Spam Data Model Complexity

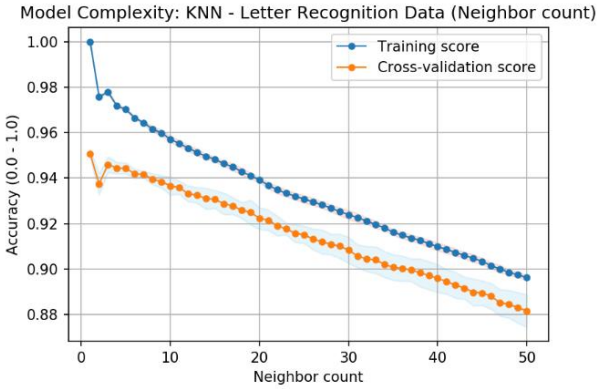

Figure 3.4 Letter Recognition Model Complexity

Artificial Neural Network (ANN) Learner

Artificial Neural Networks were constructed by tuning hyperparameters such as Relu and logistic activation functions, layer sizes, alpha values (regularization penalty), and learning rates. The neural network used was a multi-player perceptron in a feedforward network. The best performance of each activation function was highlighted below along with their associated ranks. Additional data amongst the ranks is also shown for varying layer sizes to provide a brief scope of the output data.

Table 4.1. Spam Data – ANN Hyperparameters and Scores

Rank	Activation	Layer Size	Alpha	Learning Rate (Constant)	Mean Test Score	Mean Training Score
1	Relu	(114, 114, 114)	1.00E-05	0.032	93.86%	95.71%
2	Relu	(57, 57, 57)	0.10	0.004	93.67%	95.51%
3	Logistic	(28,)	0.0001	0.128	93.62%	95.99%

Table 4.2. Letter Recognition Data – ANN Hyperparameters and Scores

Rank	Activation	Layer Size	Alpha	Learning Rate (Constant)	Mean Test Score	Mean Training Score
1	Relu	(32, 32, 32)	1.00E-05	0.008	91.77%	94.89%
2	Relu	(32, 32, 32)	3.16E-06	0.008	91.62%	94.66%
3	Relu	(32, 32, 32)	0.1000	0.008	91.23%	94.32%
11	Relu	(32, 32)	0.0316	0.016	90.76%	93.40%
89	Logistic	(32, 32, 32)	0.0100	0.032	89.51%	91.90%

Tables 4.1 and 4.2 show that the activation function did not have much effect on accuracy for the spam dataset, but in the letter recognition dataset the Relu activation function performed best. This could be the result a more sparse representation of nodes within network when using a Relu function. Since the Relu activation function hypothesis is characterized by h=max(0,a) where "a" is a linear function, it could be that non-essential nodes are rendered useless with an activation value of 0. This may increase the weight of the relevant nodes in the layer, resulting in an increase in accuracy. However, a deeper investigation of the node weights in each layer will need to be performed to confirm this.

The learning curves in Figures 4.1 and 4.2 indicate low variance and low bias within the learner. We can see that the learner was able to learn quickly relative to the total number of training examples in the data set, and the gap between the training data and the cross-validation data is marginal. The model complexity graphs in Figures 4.3 and 4.4 show that the alpha parameter did not have much effect on the accuracy of the learner until it reached a magnitude of ~10^{-1} where the higher learning rate resulted in a loss of performance. The purpose of this hyperparameter is to prevent overfitting and it penalizes nodes which perform well. However, if the penalty become too large it will result in underfitted the data. Finally, the epoch graphs in Figures 4.5 and 4.6 show that the effect of the number of epochs used to train models decreases over time. In these cases, there isn't any new information for the learner to tune activation values across epochs. This could be combatted by increasing the number of instances in the data or by adding more relevant features, but this could also come at the cost of overfitting the data.

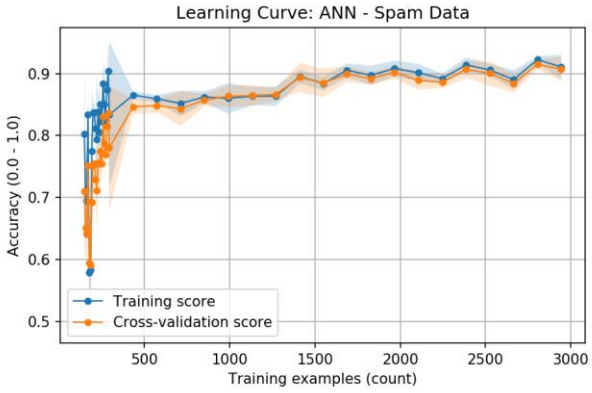

Figure 4.1 Spam Data ANN Learning Curve

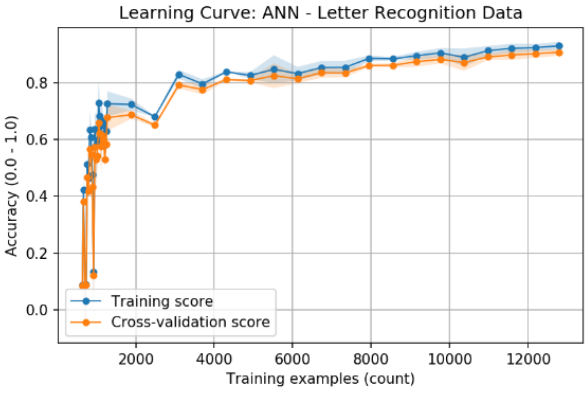

Figure 4.2 Letter Recognition ANN Learning Curve

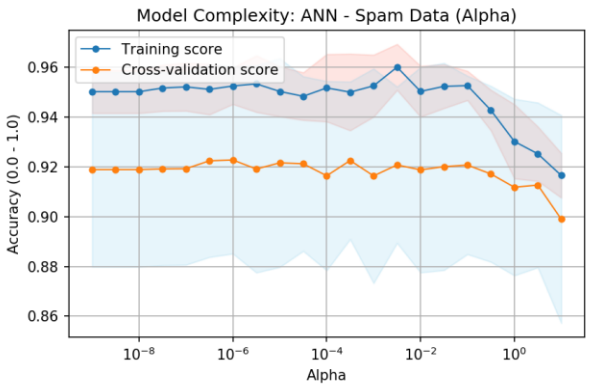

Figure 4.3 Spam Data ANN Model Complexity Curve

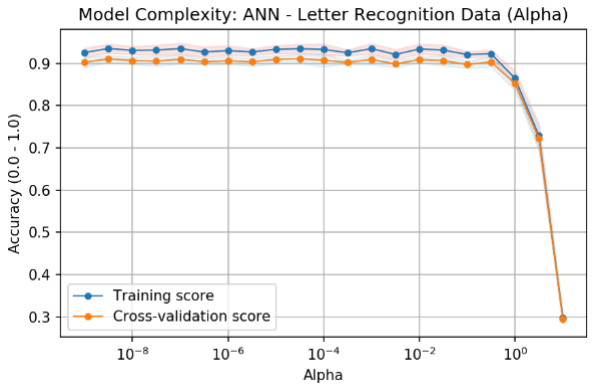

Figure 4.4 Letter Recognition ANN Model Complexity Curve

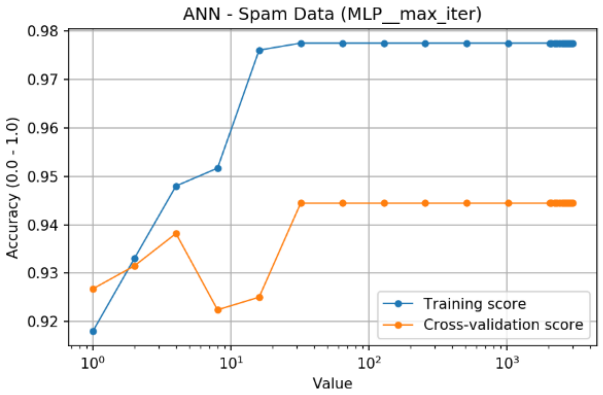

Figure 4.5 Spam Data ANN Epoch Curve

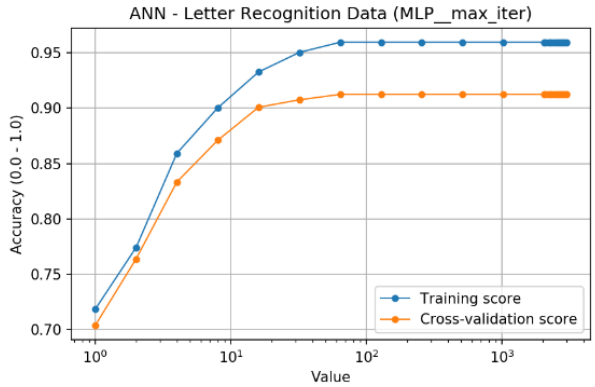

Figure 4.6 Letter Recognition ANN Epoch Curve

Support Vector Machine (SVM) Learner using the Radial Basis Function (RBF)

Support vector machines using the RBF kernel were tested with varying penalty (C), gamma, function shape (ovo or ovr), and tolerance hyperparameters. A brief overview of the outputs can be seen in Tables 5.1 and 5.2 below.

Table 5.1. Spam Data SVM-RBF Hyperparameters and Scores

Rank	C (penalty)	Gamma	Tolerance	Decision Function Shape	Mean Test Score	Mean Training Score
1	1.501	0.01754	0.0200	ovo	92.65%	94.58%
2	1.501	0.01754	0.0300	ovo	92.65%	94.57%

Table 5.2. Letter Recognition Data SVM-RBF Hyperparameters and Scores

Rank	C (penalty)	Gamma	Tolerance	Decision Function Shape	Mean Test Score	Mean Training Score
1	2.001	0.0625	0.0001	ovo	95.10%	97.03%
2	1.501	0.0625	0.0001	ovo	94.69%	96.43%

Table 5.1 shows that the SVM-RBF learner performed well on the spam data in terms of accuracy with a score of 92.65%. However, the learning curve in Figure 5.1 below suggests that the variance between the training and cross validated data was high. Generally, this is an indication of overfitting training data, but given the accuracy score the fit can be deemed acceptable in this test case. Even with this in mind, it should be noted that the model could perform worse on new test cases. Table and Figure 5.2 indicate that the SVM-RBF learner was very successfully classified the letter recognition data. Here, bias and variance were low, as demonstrated with the accuracy score of 95.10%. This could be the result of the algorithms ability to classify data of high variance between features with complex functions. We will soon see that an SVM with a linear kernel, which uses a simple linear function to classify data was not nearly as effective. The effect of increasing the penalty score had little effect on the spam data and the letter recognition data once a threshold was reached, as seen in the model complexity curves in Figures 5.2 and 5.3.

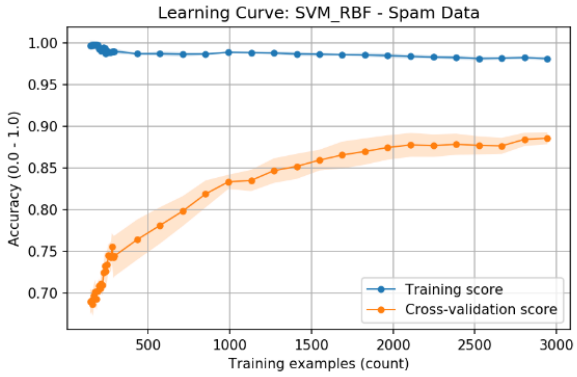

Figure 5.1 Spam Data SVM RBF Learning Curve

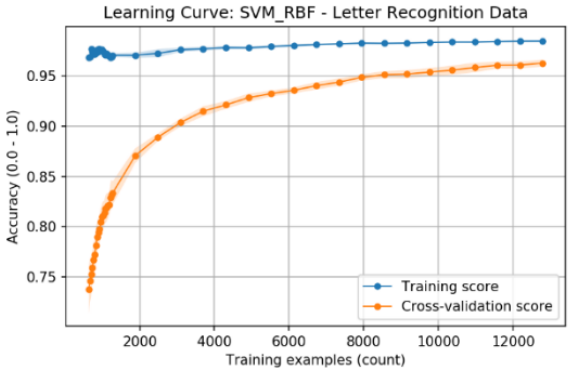

Figure 5.2 Letter Recognition Data SVM RBF Learning Curve

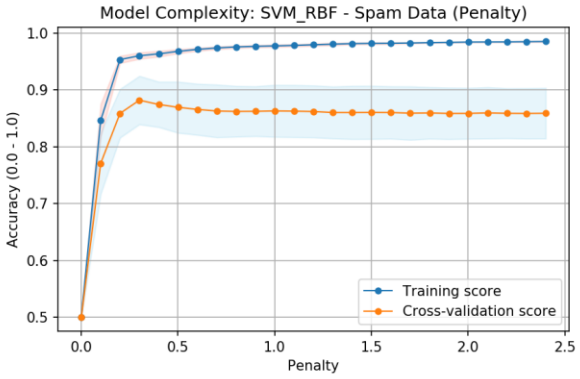

Figure 5.3 Spam Data SVM RBF Learning Curve

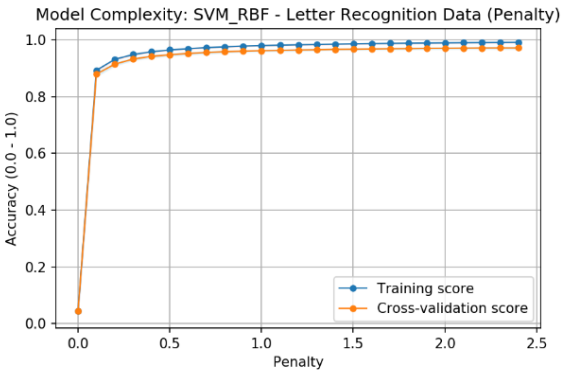

Figure 5.4 Letter Recognition Data SVM RBF Learning Curve

Support Vector Machine (SVM) Learner with no Kernel (Linear)

Linear support vector machines were constructed by tuning penalty (C) and tolerance hyperparameters. A brief overview of the output data can be seen in Tables 6.1 and 6.2 below.

Table 6.1. Spam Data Linear SVM Hyperparameters and Scores

Rank	C (penalty)	Tolerance	Mean Test Score	Mean Training Score
1	1.501	0.0200	92.31%	92.88%
2	1.001	0.0200	92.08%	92.94%

Table 6.2. Letter Recognition Data Linear SVM Hyperparameters and Scores

Rank	C (penalty)	Tolerance	Mean Test Score	Mean Training Score
1	2.001	1.00E-08	70.52%	70.96%
2	1.501	0.0010	69.70%	70.31%

Table 6.1 suggests that the test score accuracy on the spam dataset was comparable to the other learners previously discussed. From the learning curves in Figures 6.1 and 6.2, it was clear that as the number of training examples increased, it became more difficult to classify the training data. Eventually, the learning curve and cross validation curves converged, finding the appropriate hyperplanes to classify the given data. Table 6.2 expresses that the linear SVM kernel did not perform well for the letter recognition data, scoring 70.96% in accuracy. As mentioned earlier, this is likely the result of the linear SVM algorithm's in ability to classify complex data with higher variance with a simple linear kernel.

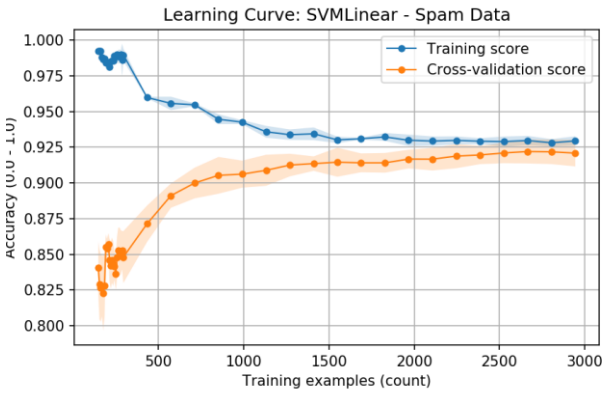

Figure 6.1 Spam Data SVM Linear Learning Curve

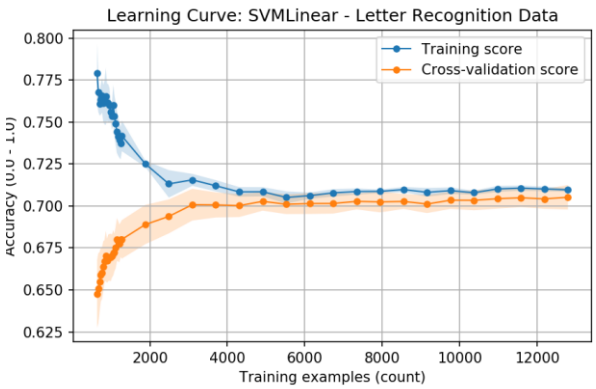

Figure 6.2 Letter Recognition Data SVM Linear Learning Curve

Comparisons

A comparison between learners in the spam dataset was carried using score time, fit time, and accuracy metrics. The results are shown below in Tables 7.1.

Table 7.1. Spam Data Performance Metrics Across Learners

Spam Dataset			
Learner	**Score Time (s)**	**Fit Time (s)**	**Test Accuracy**
Decision Tree	0.0008	0.0828	91.29%
AdaBoost DT	0.0114	3.3600	94.33%
KNN	0.1077	0.0142	89.88%
ANN	0.0038	0.3547	93.86%
SVM-Linear	0.0431	0.2366	92.88%

As seen from the table above, each learner performed well regardless of the unbalanced ratio of the high number of features relative to the number of samples. Using accuracy as the sole metric for performance in the spam dataset, the best learner would be the Adaboost decision tree, as it resulted in an accuracy of 94.33%. However, it should be noted that the algorithm introduces a tradeoff. While the learner had the best accuracy, it also had the longest fit time by an order of magnitude from the next best classifier. If fit time is a metric important to the user, one may compromise on this trade off by using an ANN or a standard DT learner.

The same comparison on the spam dataset was done to the letter recognition dataset. A summary of the results can be seen below in Table 7.2.

Table 7.2. Letter Recognition Data Performance Metrics Across Learners

Letter Recognition Dataset			
Learner	**Score Time (s)**	**Fit Time (s)**	**Test Accuracy**
Decision Tree	0.0022	0.9590	82.74%
AdaBoost DT	0.0580	37.8780	93.68%
KNN	0.2439	0.0525	94.53%
ANN	0.0072	1.3903	91.77%
SVM-RBF	1.2495	2.3344	95.10%

As seen in the performance metrics above, there was no overbearing complications with features being derived from the same pixel information. Using accuracy as the sole performance metric, it can be concluded that the SVM-RBF algorithm performed the best. However, for a small trade-off in accuracy and a significant saving in score and fit times, one may be more inclined to use the KNN algorithm to classify this dataset.

References

Hopkins, M., Reeber, E., Foreman, G., & Suermondt, J. (1999, January 07). Retrieved February 2, 2019, from https://archive.ics.uci.edu/ml/datasets/spambase

Slate, D. J. (1991, January 01). Retrieved February 03, 2019, from https://archive.ics.uci.edu/ml/datasets/Letter Recognition